МГТУ им. Н. Э. Баумана

ИУ 7 – 32

Отчет о лабораторной работе №5

«Обработка очередей»

Вариант №8

Исаев Д.С.

# Цель работы

# Отработка навыков работы с типом данных «очередь», представленным в виде одномерного массива и односвязного линейного списка. Сравнительный анализ реализации алгоритмов включения и исключения элементов из очереди при использовании указанных структур данных. Оценка эффективности программы по времени и по используемому объему памяти.

# Условие задачи

Система массового обслуживания состоит из обслуживающего аппарата (ОА) и очереди заявок.

**T1** **T2**

OA

Заявки поступают в "хвост" очереди по случайному закону с интервалом времени **Т1,** равномерно распределенным от **0 до 6** единиц времени (е.в.). В ОА они поступают из "головы" очереди по одной и обслуживаются также равновероятно за время **Т2** от **0 до 1** е.в., Каждая заявка после ОА вновь поступает в "хвост" очереди, совершая всего 5 циклов обслуживания, после чего покидает систему. (Все времена – **вещественного** типа) В начале процесса в системе заявок нет.

Смоделировать процесс обслуживания до ухода из системы первых 1000 заявок, выдавая после обслуживания каждых 100 заявок информацию о текущей и средней длине очереди, а в конце процесса - общее время моделирования и количества вошедших в систему и вышедших из нее заявок, количестве срабатываний ОА, время простоя аппарата. По требованию пользователя выдать на экран адресов элементов очереди при удалении и добавлении элементов. Проследить, возникает ли при этом фрагментация памяти.

**Описание**

**Очередь**

Очередь – это последовательный список переменной длины, включение элементов в который идет с одной стороны (с «хвоста»), а исключение – с другой стороны (с «головы»). Принцип работы очереди: первым пришел – первым вышел, т. е. First In – First Out (FIFO).

Для «хвоста» и «головы» очереди используют соответственно два указателя Pin и Pout,. то есть, исключается элемент с адресом Pout и включается элемент по адресу Pin. Моделировать простейшую линейную очередь можно как на основе вектора (одномерного массива), так и на основе динамического списка.

При просмотре очереди (классически) ее элемент должен вытолкнуться.

На практике реализуется и просмотр и выталкивание верхнего элемента.

**Реализация**

**Массив**

При моделировании простейшей линейной очереди на основе одномерного массива выделяется последовательная область памяти из m мест по L байт, где L – размер поля данных для одного элемента размещаемого типа. В каждый текущий момент времени выделенная память может быть вся свободна, занята частично или занята полностью.

В начале процесса очередь пуста, при этом: Pin = Pout =Q1, где Q1 – адрес (индекс) первого элемента очереди, а количество элементов очереди (K) равно нулю. При включении очередного элемента в очередь он располагается по адресу Pin, а сам указатель Pin перемещается на длину типа данных к началу следующего элемента (рис. 1). На рисунке символом Х обозначена память, занятая элементами очереди, при этом считаем, что под массив была выделена область памяти от Q1 до Qm.

Q1 Q2 Q**m**

… Х Х Х

Pout Pin

**Рис. 1**

В очереди исключается элемент, находящийся по адресу (индексу) Pout, а указатель Pout, как и в случае добавления, перемещается к следующему элементу. Переполнение очереди наступает, когда указатель «хвоста» Pin перейдет границу последнего элемента, т. е. возникает ситуация, при которой Pin = Qm+L

**Список**

Большинство проблем, возникающих при реализации очереди в виде массива, устраняется при реализации очереди на основе односвязного линейного списка, каждый элемент которого содержит информационное поле и поле с указателем «вперед» (на следующий элемент). В этом случае в статической памяти можно либо хранить адрес начала и конца очереди, либо – адрес начала очереди и количество элементов. Исходное состояние очереди: Pout = Pin - пустой указатель, K = 0.

Основные операции над очередью: включение элемента в очередь и исключение элемента из очереди.

Память под элементы списка освобождается посредством системных функций, за выполнение которых ответственна ОС.

**Результаты**

В результате работы программы были получены следующие данные:

**Реализация массивом:**

* Вошло заявок: 1006
* Вышло заявок: 1000
* Автомат сработал 5008 раз
* Время моделирования: 3031 е.в.
* Время простоя: 477.3 е.в.
* Погрешность (по входу): 0.43%
* Погрешность (по выходу): 1.7%
* Погрешность (по времени): 1%
* Затрачено памяти: 262144 байт

**Реализация списком:**

* Вошло заявок: 1007
* Вышло заявок: 1000
* Автомат сработал 5016 раз
* Время моделирования: 2994 е.в.
* Время простоя: 461.8 е.в.
* Погрешность (по входу): 0.91%
* Погрешность (по выходу): 0.8%
* Погрешность (по времени): 0.21%
* Затрачено памяти: 80 байт

В результате наблюдения за освобождаемой памятью выяснилось, что освободившаяся память почти всегда сразу же повторно используется при выделении памяти, что очень хорошо и не способствует фрагментации памяти – т.е. тому что непрерывные о логике данные располагаются в разных участках памяти.

Результаты показывают, что использование массива для поставленной задачи менее эффективно по памяти, так как, несмотря на то, что массив динамический, память в нем выделяется порциями.

К тому же реализация списком позволяет избежать проблем с переполнение и усложнения структуры кольцевой очередью.

Использование массивов позволяет получить выигрыш в производительности. Если данные в очереди часто изменяются, то лучше применить реализацию списком, иначе – массивом.

**Вывод:**

Из результатов работы программы видно, что в данной технической задаче выгоднее использовать список для реализации очереди вместо списка, так как это более экономно по памяти, а быстродейтсвие в данной программе не критично.